Leiden University Medical Center

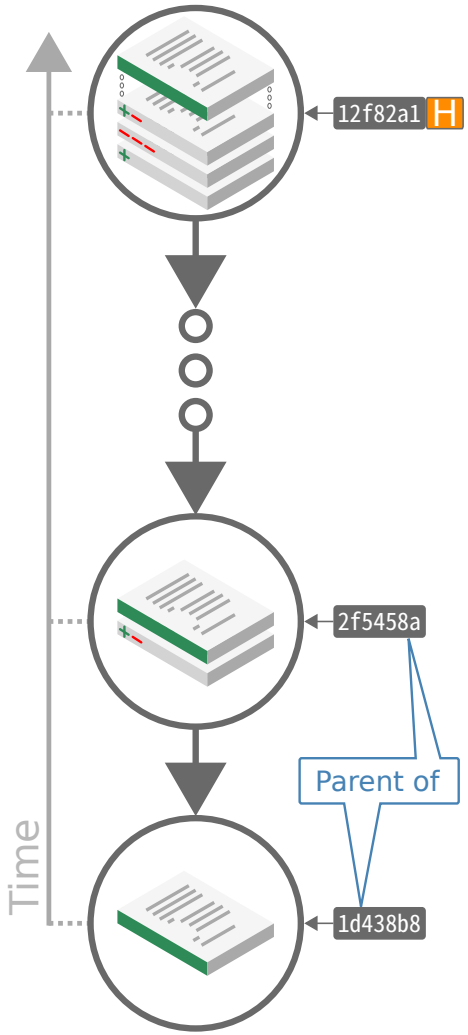
Code and data management with Git

The git commit graph

Table of contents

The git commit graph

A linear history



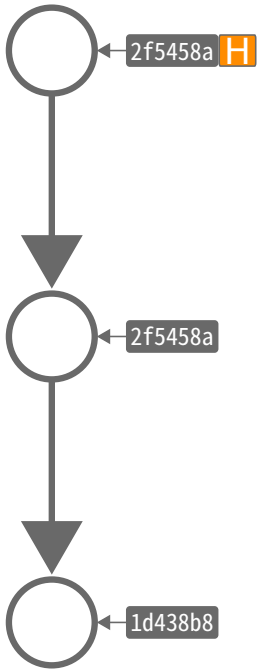
Committing creates a new commit on top of current one.

Parent of

Every commit has a parent.

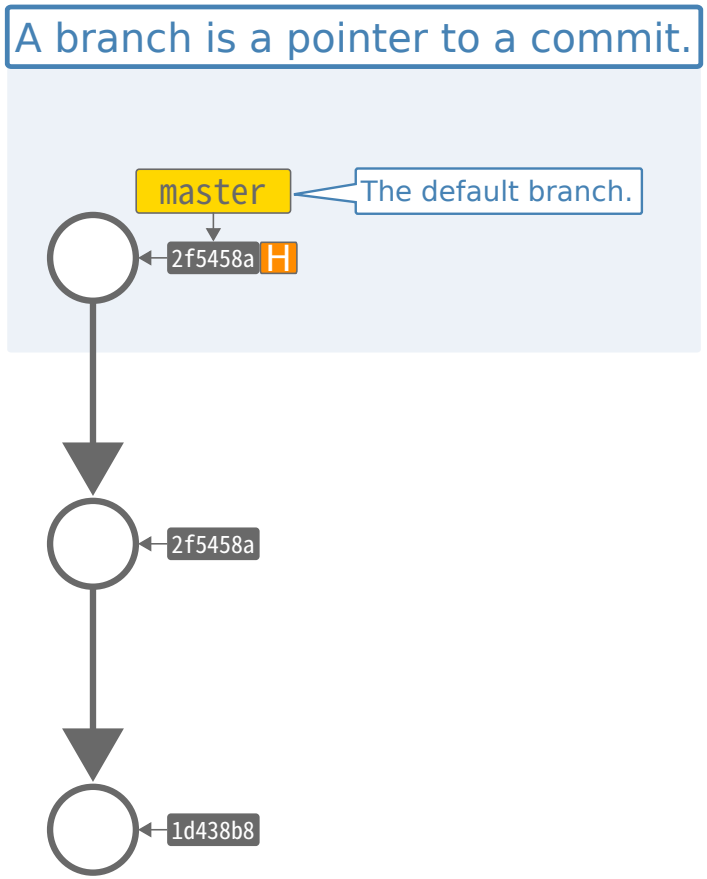
The git commit graph

A linear history

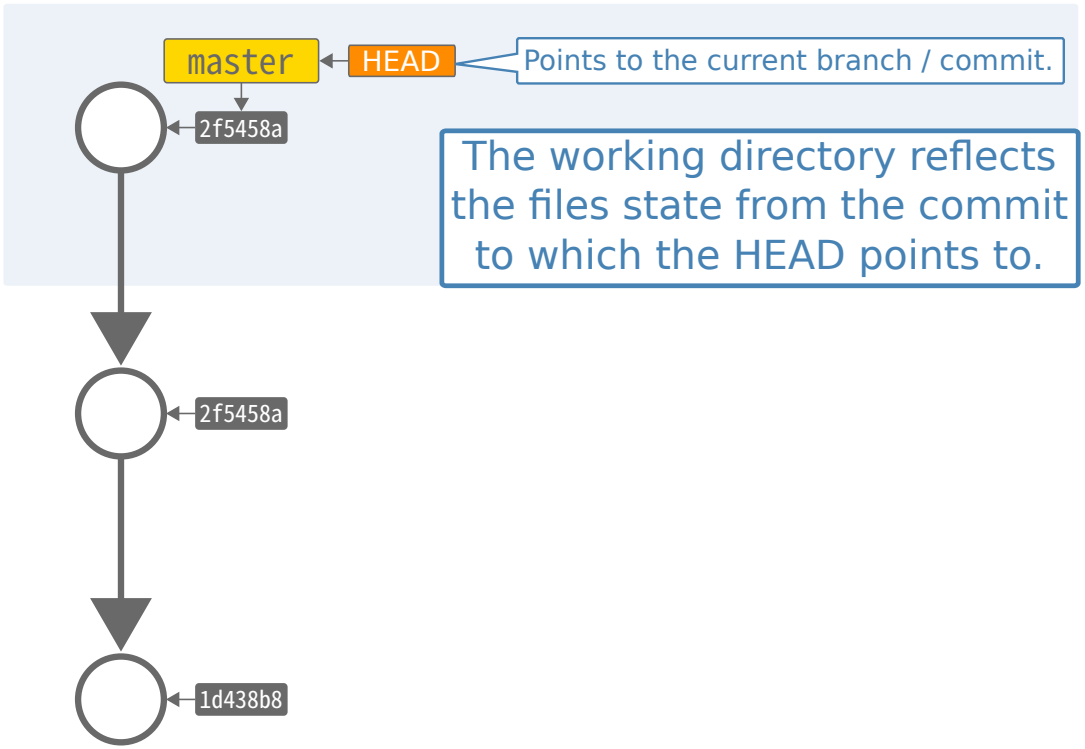


The git commit graph

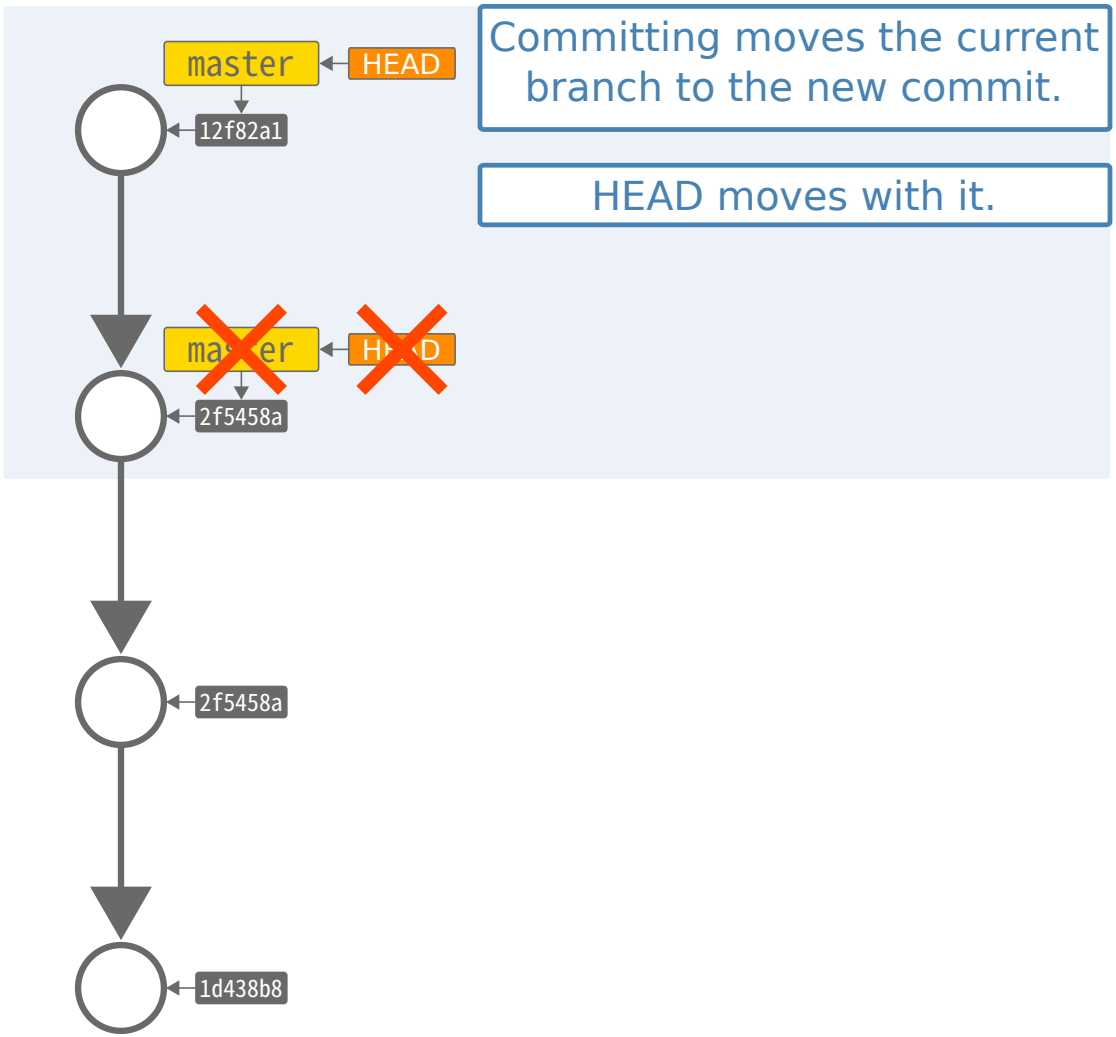
The default `master` branch



The `HEAD` pointer

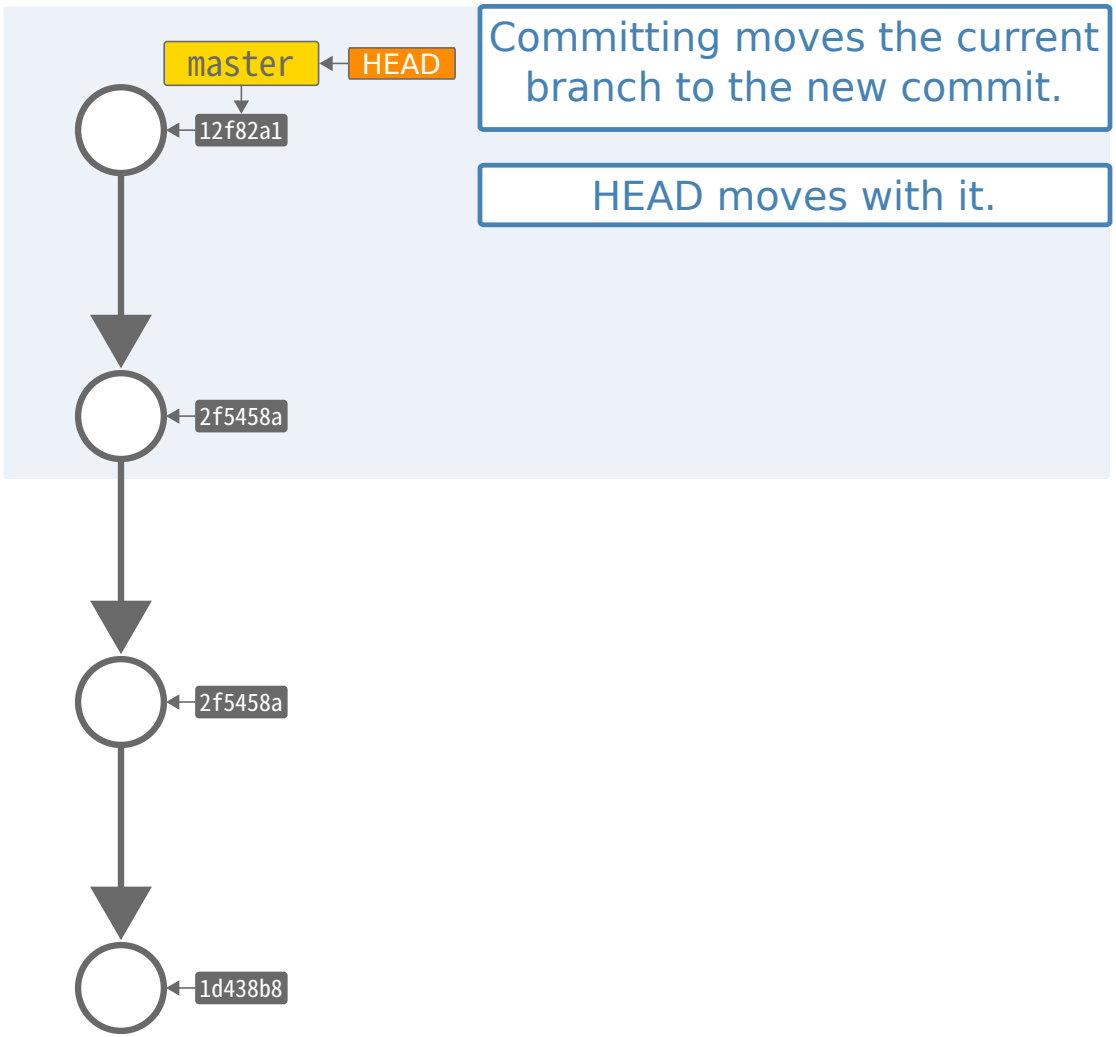


Committing


master
HEAD
12f82a1
2f5458a
2f5458a
1d438b8
Committing moves the current branch to the new commit.
HEAD moves with it.

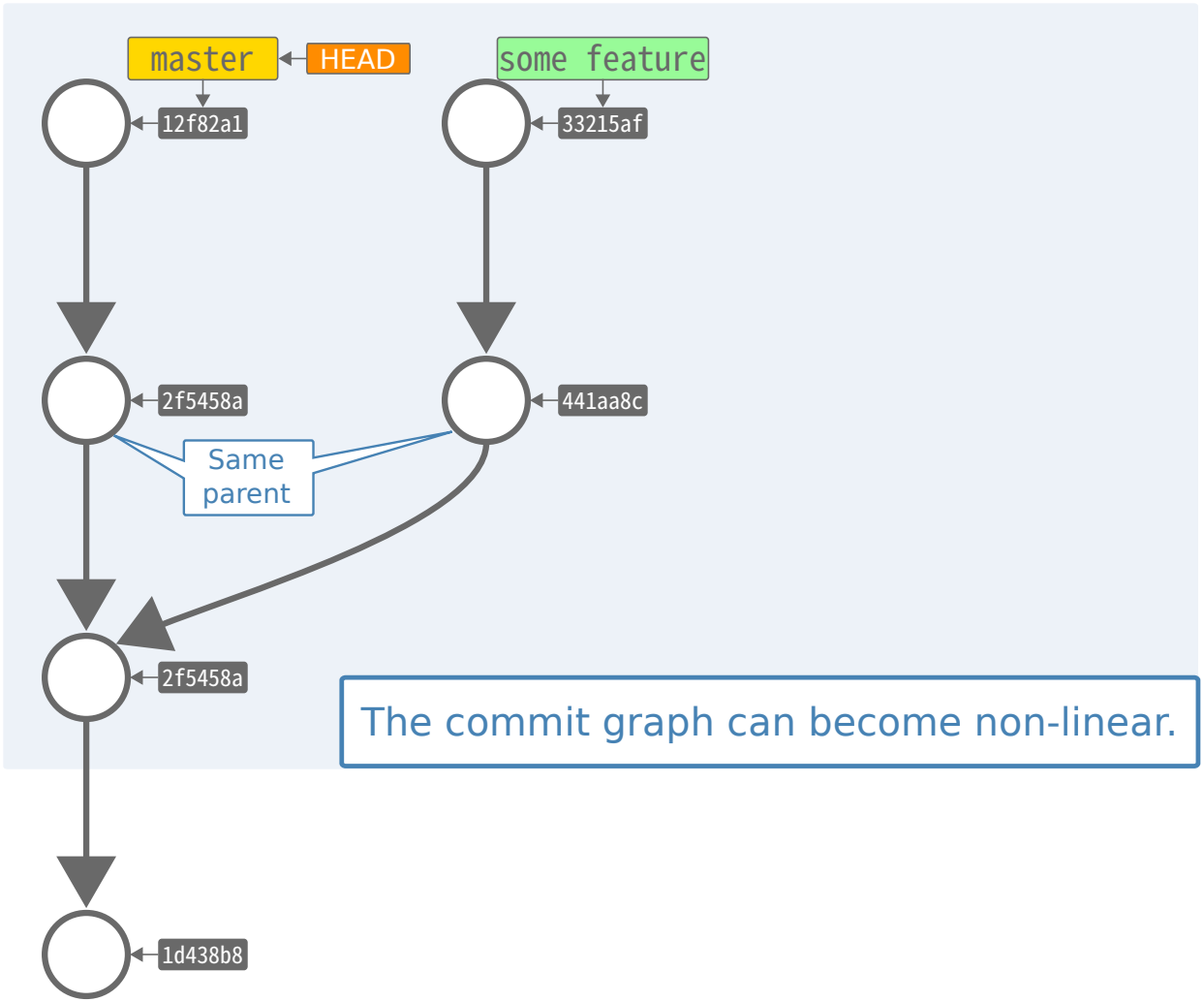
The git commit graph

Committing



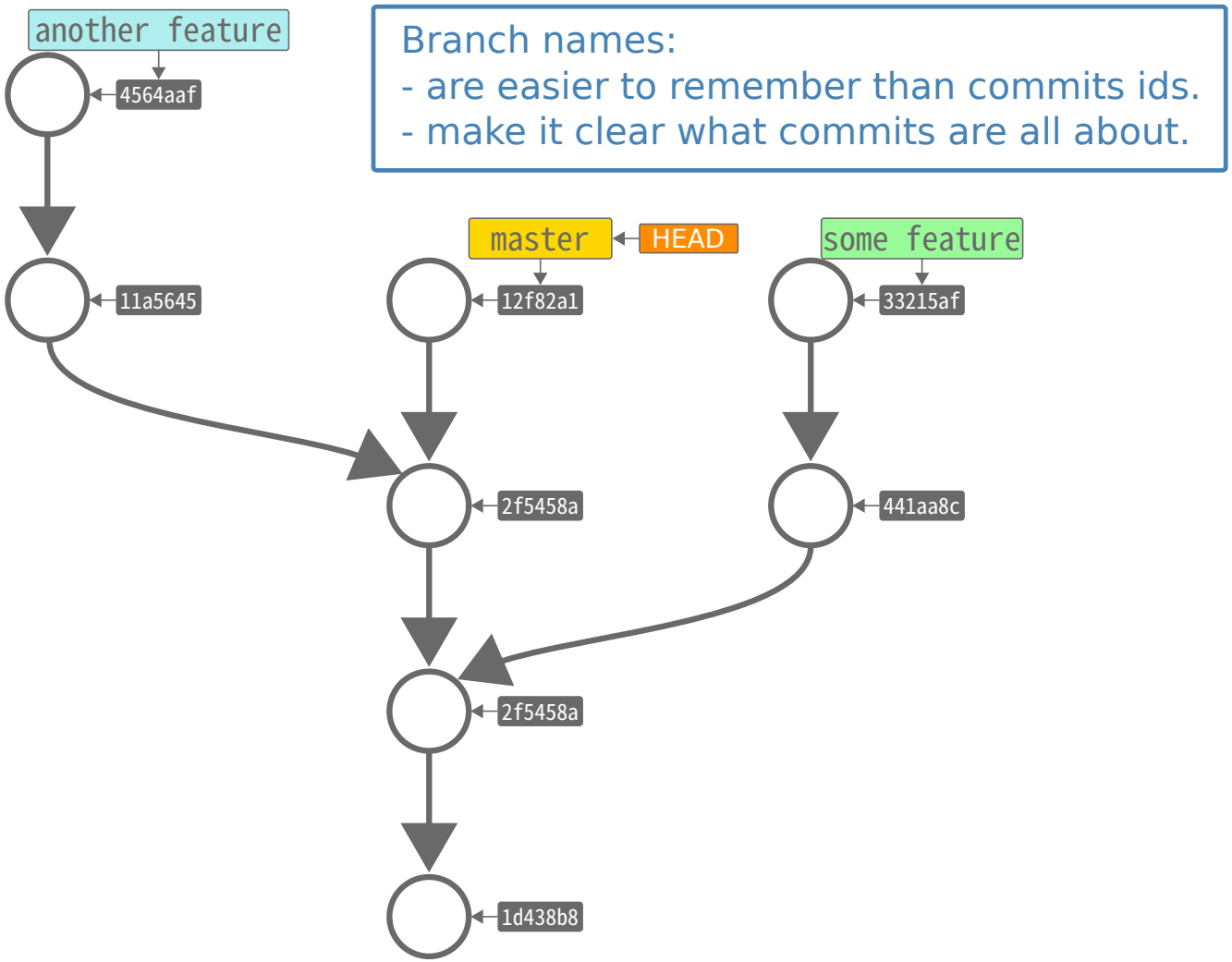
The git commit graph

A non-linear history



The git commit graph

Use branches to keep track of different code paths



Merging

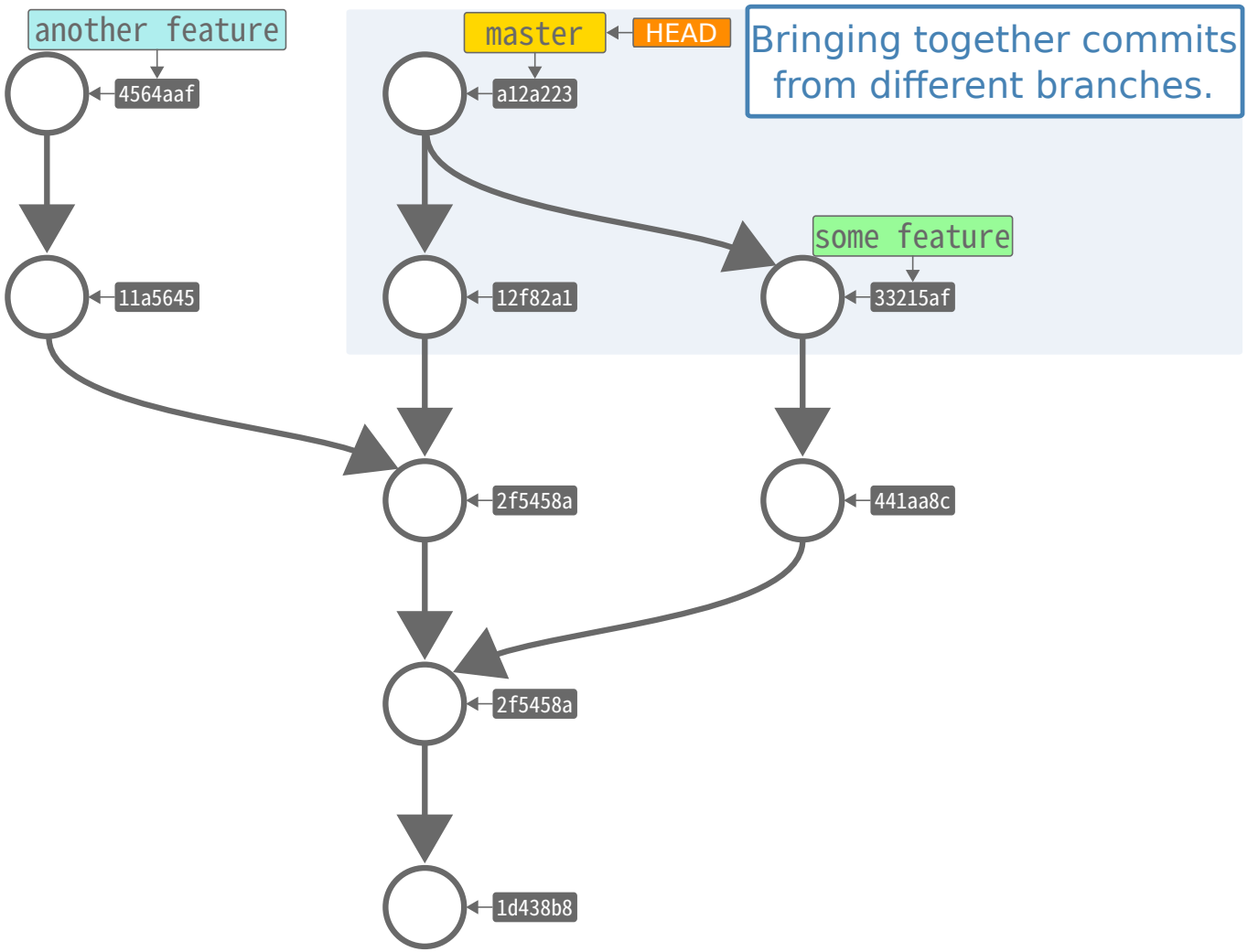
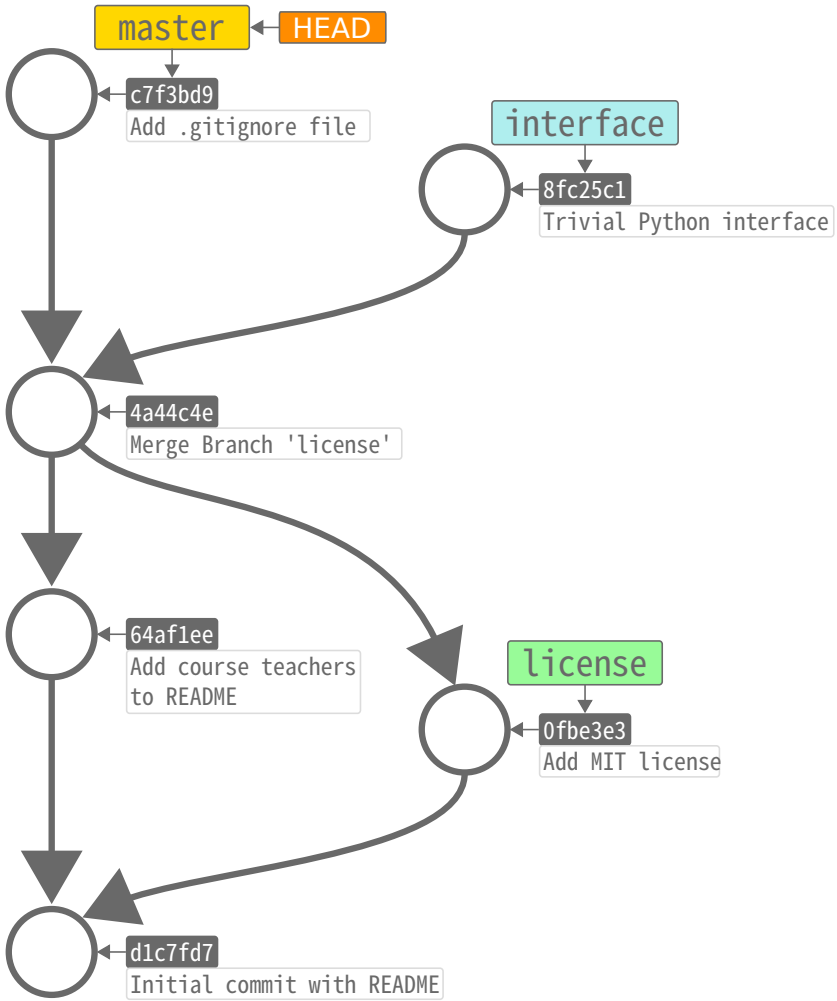

another feature
4564aaf
master
HEAD
a12a223
Bringing together commits from different branches.
some feature
11a5645
12f82a1
33215af
2f5458a
441aa8c
2f5458a
1d438b8

Table of contents

Inspecting the commit graph

An example repository



Showing the current branch

```
$ git status
On branch master
nothing to commit, working directory clean.
```

Remember: you cannot type `git status` enough!

Listing branches

```
$ git branch
  interface
  license
* master
```

Three branches, current branch is `master`.

Listing branches

```
$ git branch
  interface
  license
* master
```

Three branches, current branch is `master`.

```
$ git branch -v
  interface 8fc25c1 Trivial Python interface
  license   0fbe3e3 Add MIT license
* master    c7f3bd9 Add .gitignore file
```

`-v`: Shows the commit each branch points to.

The commit log

```
$ git log --oneline --decorate
c7f3bd9 (HEAD -> master) Add .gitignore file
4a44c4e Merge branch 'license'
64af1ee Add course teachers to README
0fbe3e3 (license) Add MIT license
d1c7fd7 Initial commit with README
```

`--oneline`: Shows commit summary on one line.
`--decorate`: Adds branch information.

Inspecting the commit graph

The commit log as a graph

```
$ git log --oneline --decorate --graph --all
* 8fc25c1 (interface) Trivial Python interface
| * c7f3bd9 (HEAD -> master) Add .gitignore file
|/
* 4a44c4e Merge branch 'license'
|\
| * 0fbe3e3 (license) Add MIT license
* | 64af1ee Add course teachers to README
|/
* d1c7fd7 Initial commit with README
```

`--graph`: Shows the commit graph.

`--all`: Includes all branches instead of just the current one.

Table of contents

Switching to another branch

```
$ git checkout interface
Switched to branch 'interface'
```

- Points HEAD to the named branch.
- Updates your working directory.

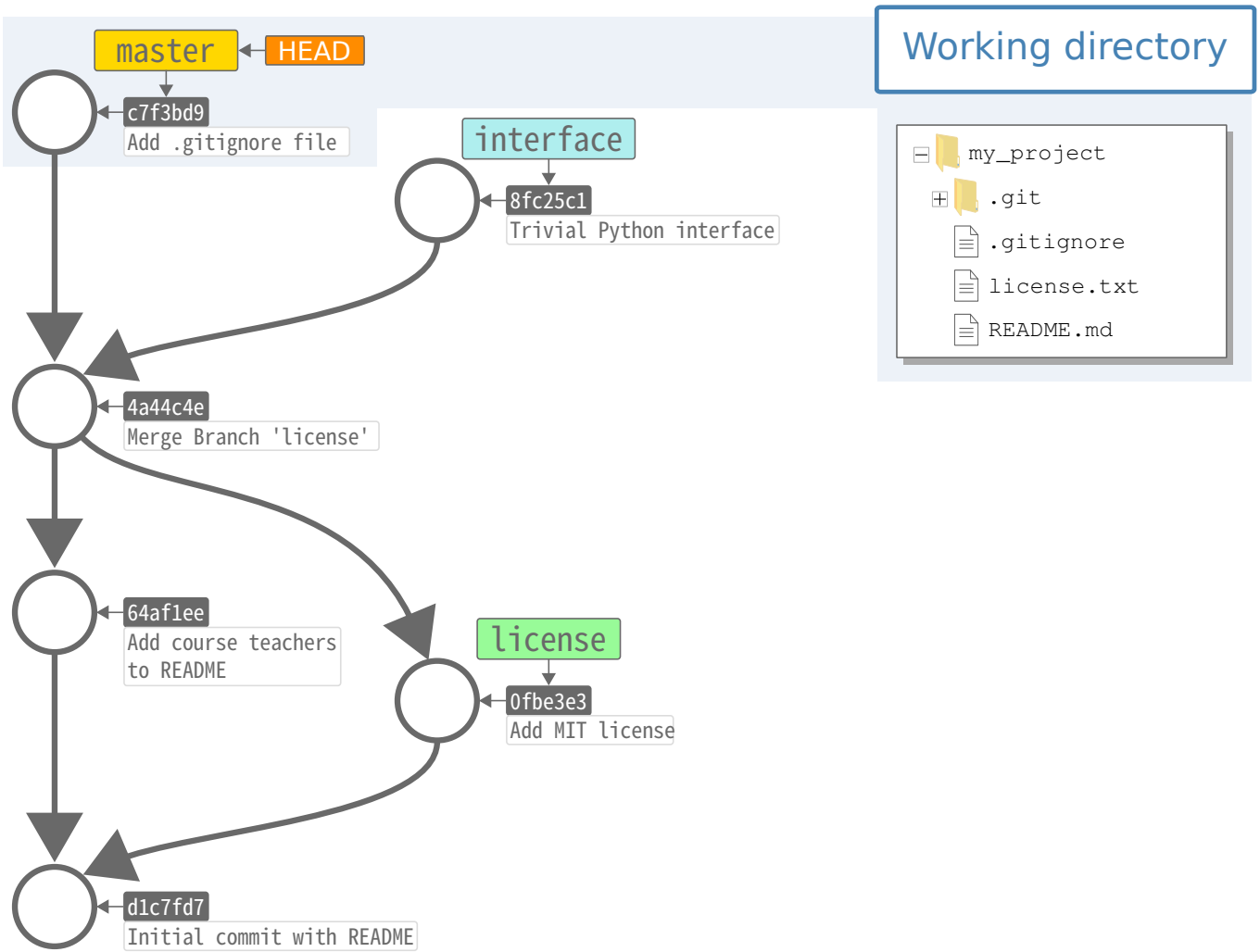
Switching to another branch

```
$ git checkout interface
Switched to branch 'interface'
```

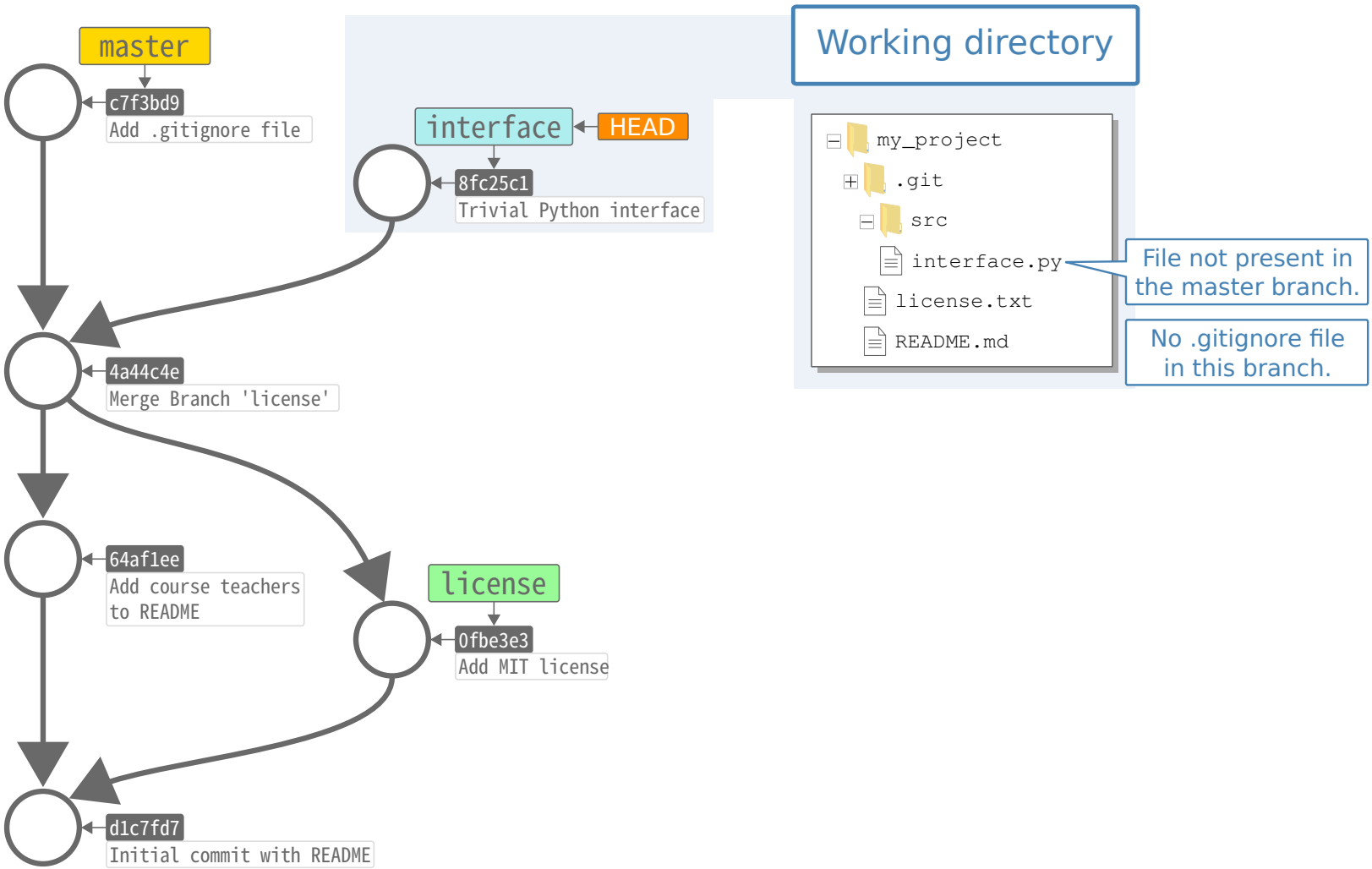
- Points HEAD to the named branch.
- Updates your working directory.

```
$ git status
On branch interface
nothing to commit, working directory clean.
```

Switching to another branch



Switching to another branch



Detached HEAD state

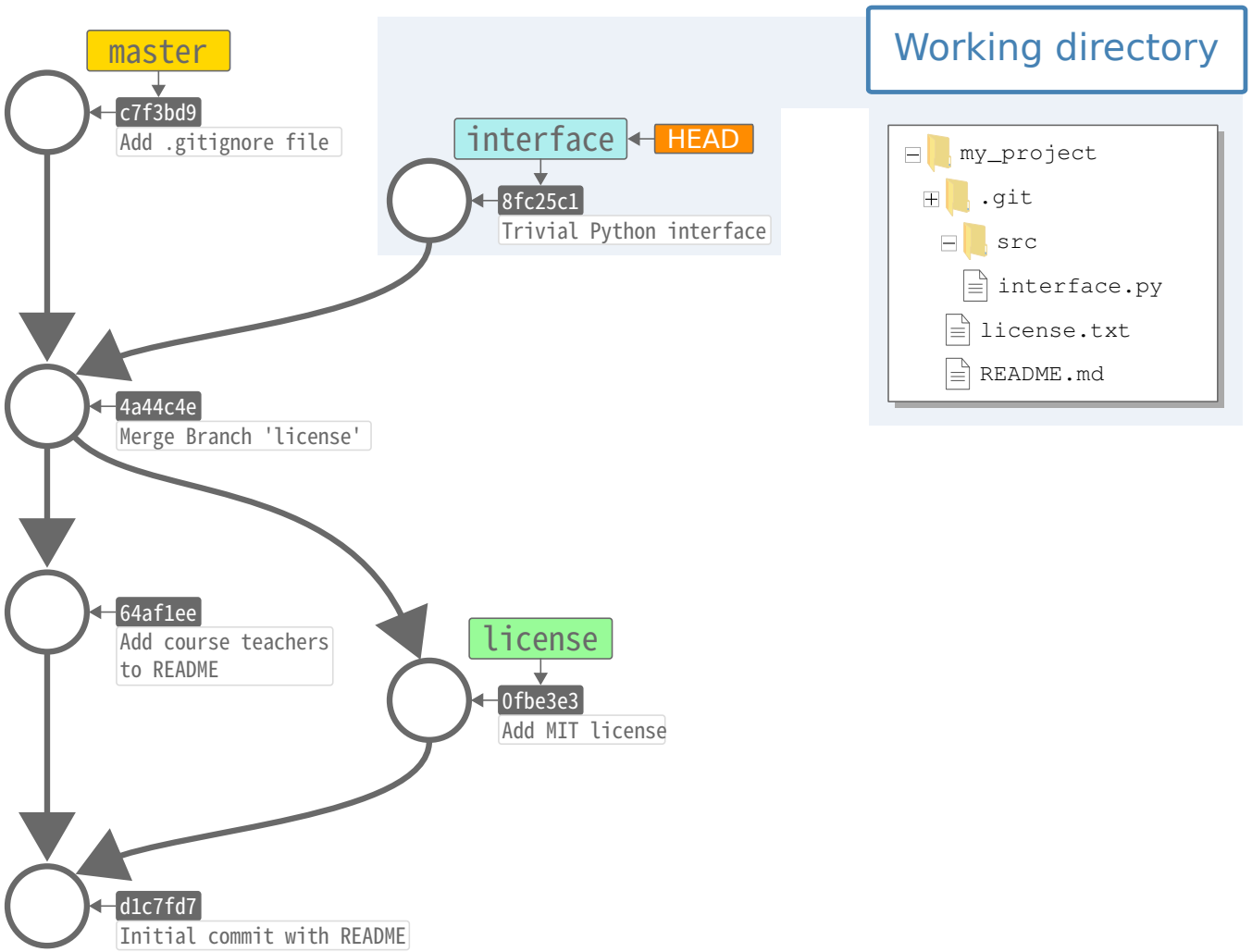
```
$ git checkout 4a44c4e
Note: checking out '4a44c4e'.

You are in 'detached HEAD' state. You can look ...

If you want to create a new branch to retain
commits you create, you may do so (now or later) by
using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 4a44c4e... Merge Branch 'license'
```

Detached HEAD state

```
$ git status
HEAD detached at 4a44c4e
nothing to commit, working directory clean.
```

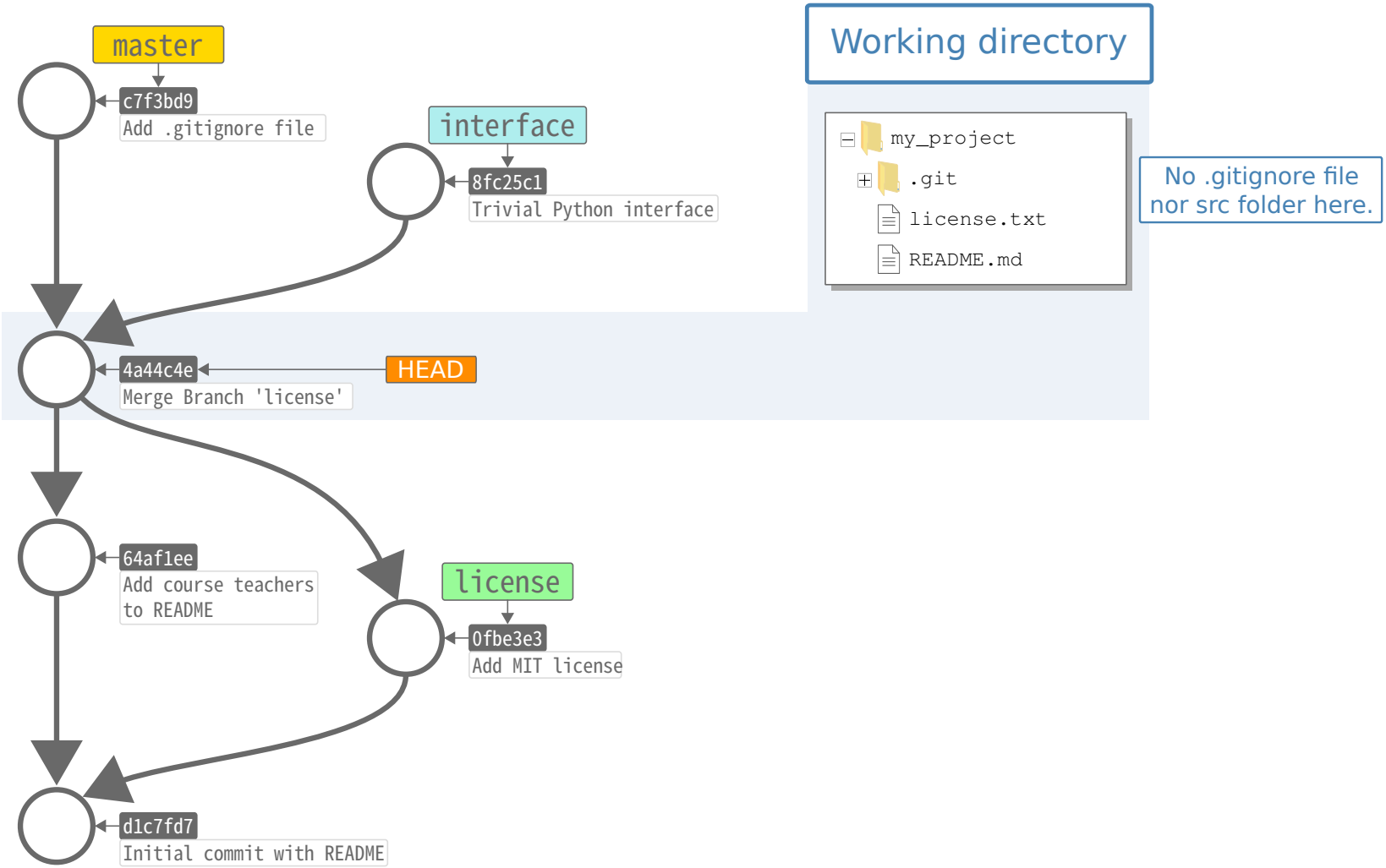
Navigating the commit graph

Detached `HEAD` state



Navigating the commit graph

Detached `HEAD` state



Navigating the commit graph

Detached `HEAD` state

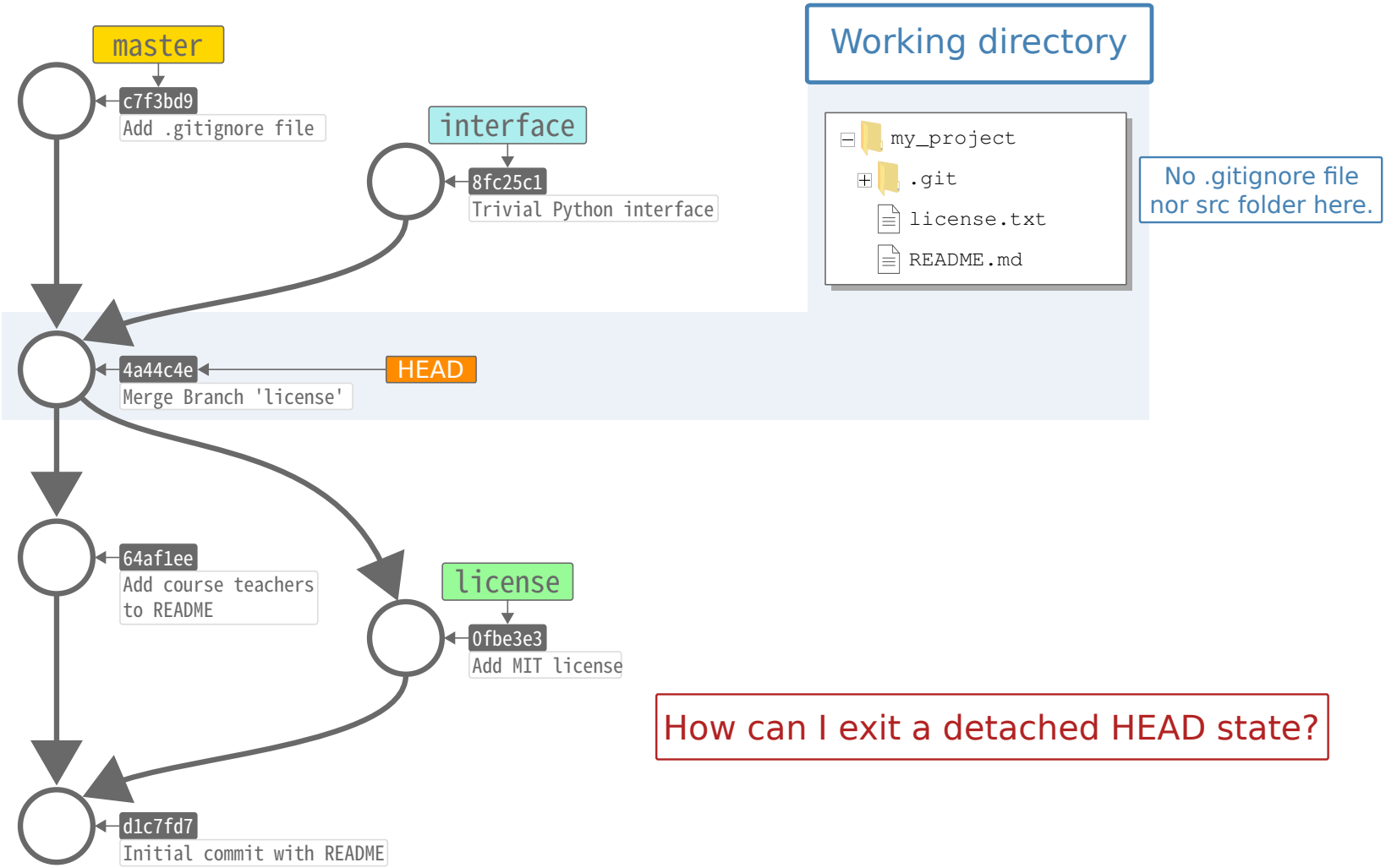


Table of contents

Creating a new branch

```
$ git branch lib
```

Creates branch `lib` at `HEAD`.

Creating a new branch

```
$ git branch lib
```

Creates branch `lib` at `HEAD`.

How can I see that it worked?

Creating a new branch

```
$ git branch lib
```

Creates branch `lib` at `HEAD`.

How can I see that it worked?

```
$ git branch
  interface
  lib
  license
* master
```

Creating a new branch

```
$ git branch lib
```

Creates branch `lib` at `HEAD`.

How can I see that it worked?

```
$ git branch
  interface
  lib
  license
* master
```

How can I switch to the new `lib` branch?

Creating a new branch

```
$ git checkout lib
Switched to branch 'lib'
```

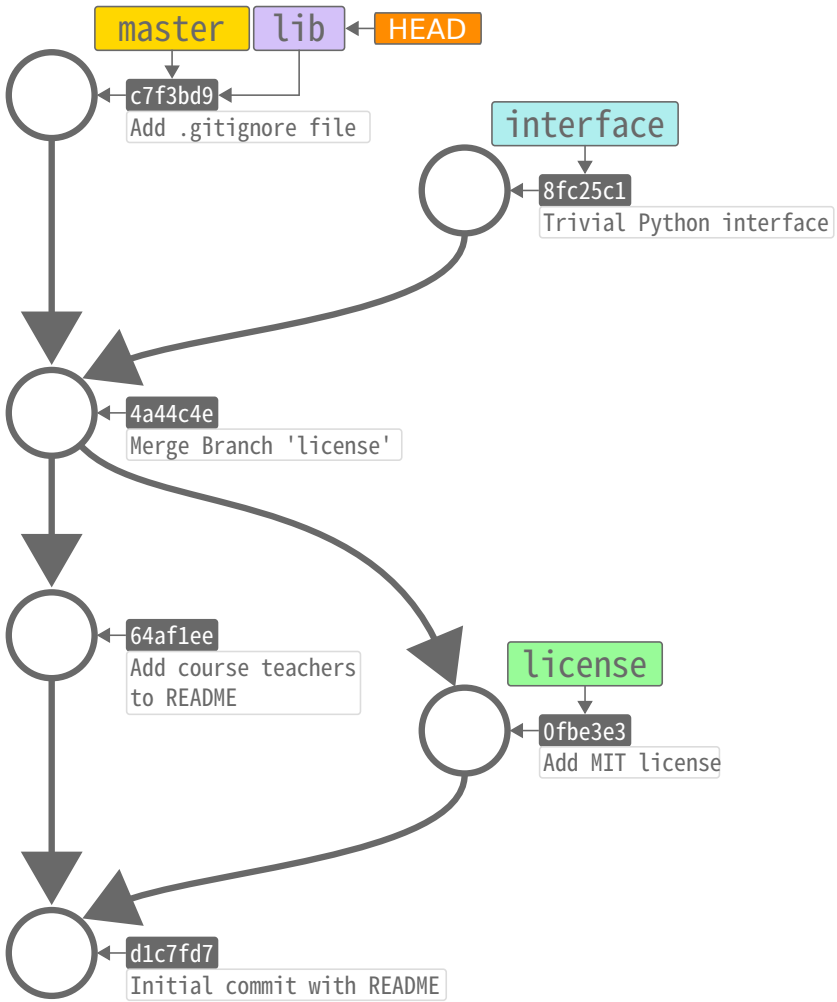
Creating a new branch

For Git pro's, in one command:

```
$ git checkout -b lib
Switched to a new branch 'lib'
```

At this point, `lib` is just a label to the same commit as `master`.

Creating a new branch



Working on a branch

We can extend the `lib` branch by adding commits.

```
$ echo 'VERSION=1' > testlib.py
$ git add testlib.py
$ git commit -m 'Add empty testlib'
```

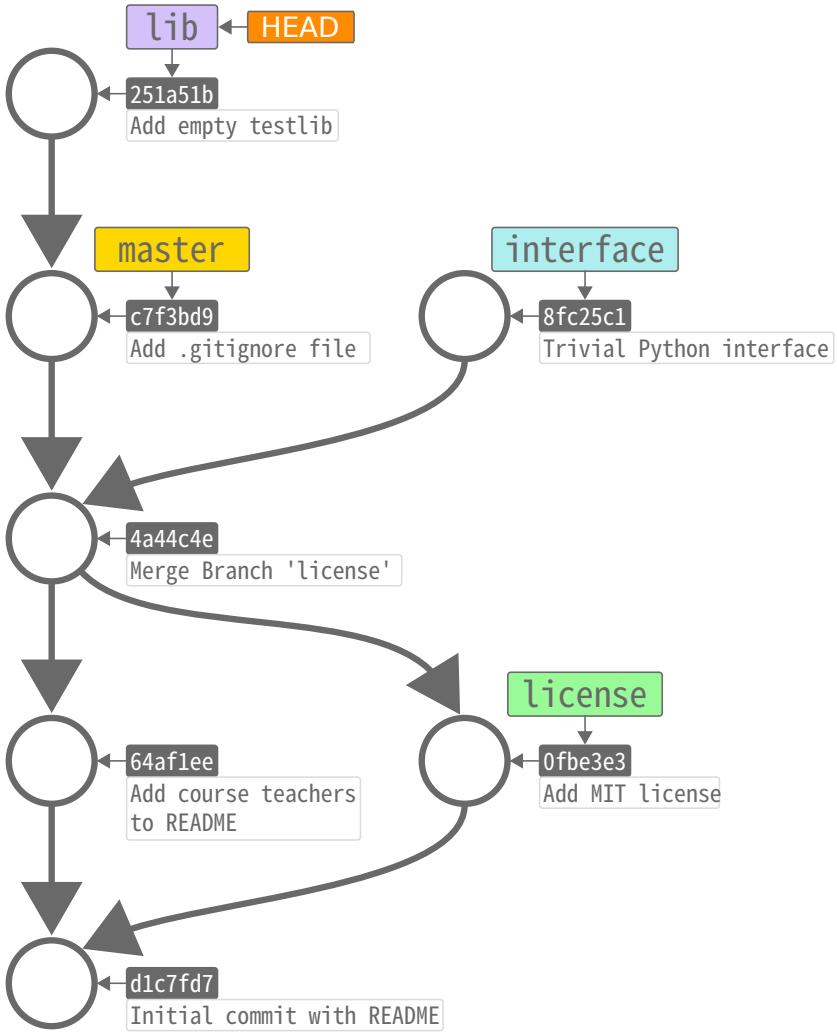
Working on a branch

We can extend the `lib` branch by adding commits.

```
$ echo 'VERSION=1' > testlib.py
$ git add testlib.py
$ git commit -m 'Add empty testlib'
```

```
$ git log --oneline --decorate --graph --all
* 251a51b (HEAD -> lib) Add empty testlib
* c7f3bd9 (master ) Add .gitignore file
| * 8fc25c1 (interface) Trivial Python interface
|/
* 4a44c4e Merge branch 'license'
...
```

Extended `lib` branch



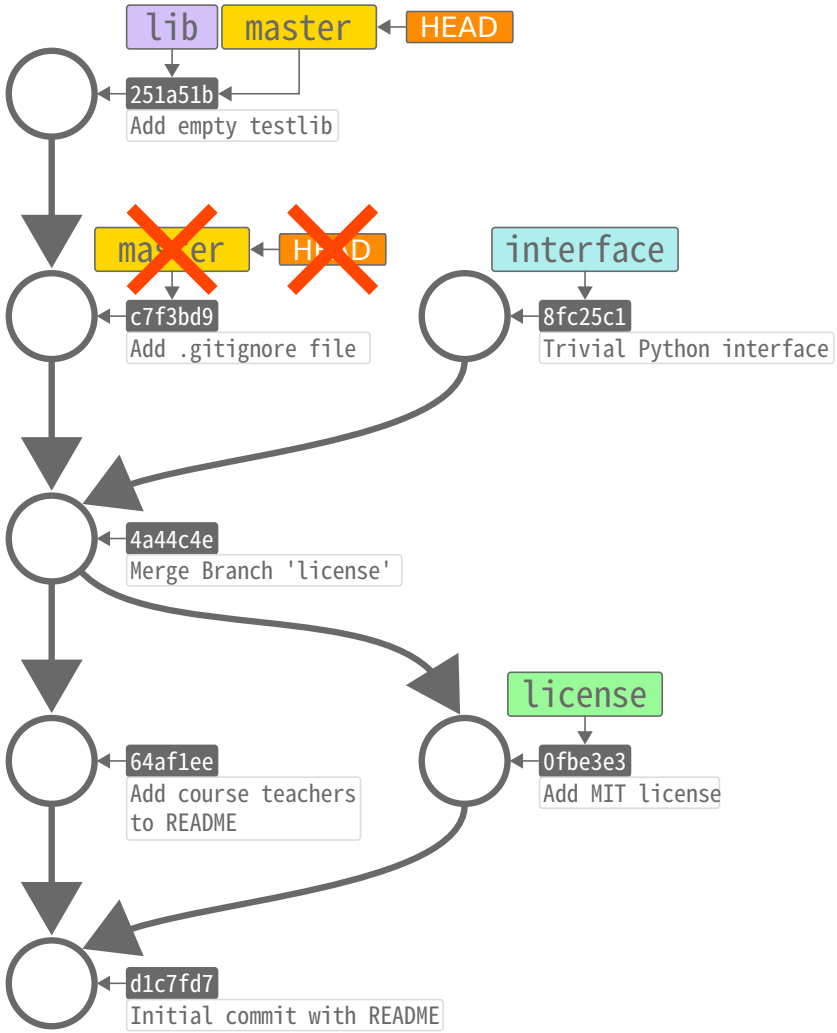
Fast forward merging

Let's merge our work on `lib` back into `master`.

```
$ git checkout master
Switched to branch 'master'
$ git merge lib
Updating c7f3bd9..251a51b
Fast-forward
 testlib.py |    1 +
 1 file changed, 1 insertion(+)
 create mode 100644 testlib.py
```

Manipulating the commit graph

Merged branch `lib` into `master` (fast forward).



Three-way merging

Let's merge our work on `interface` back into `master`.

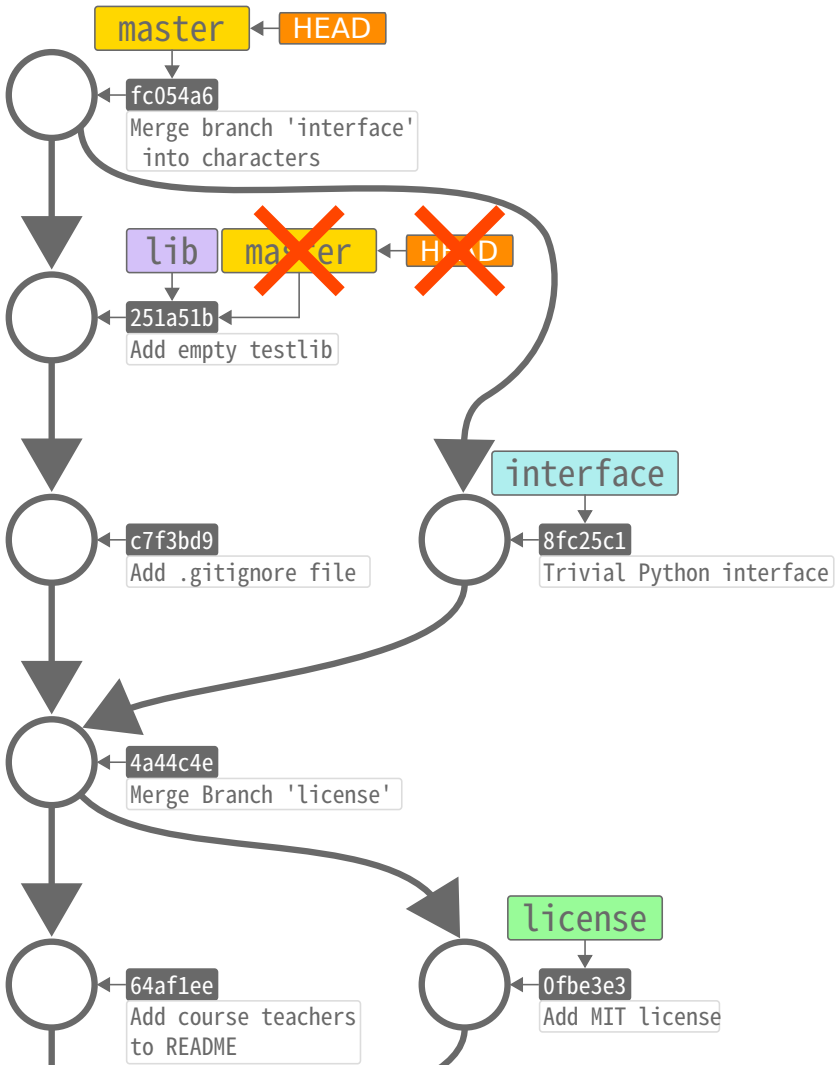
```
$ git merge interface
Merge made by the 'recursive' strategy.
 interface.py |    3 +++
 1 file changed, 3 insertions(+)
 create mode 100755 interface.py
```

This merge was harder:

- `interface` and `master` had diverged.
- Git determines changes in `interface` and `master` since their most recent common ancestor.
- A new commit is created.

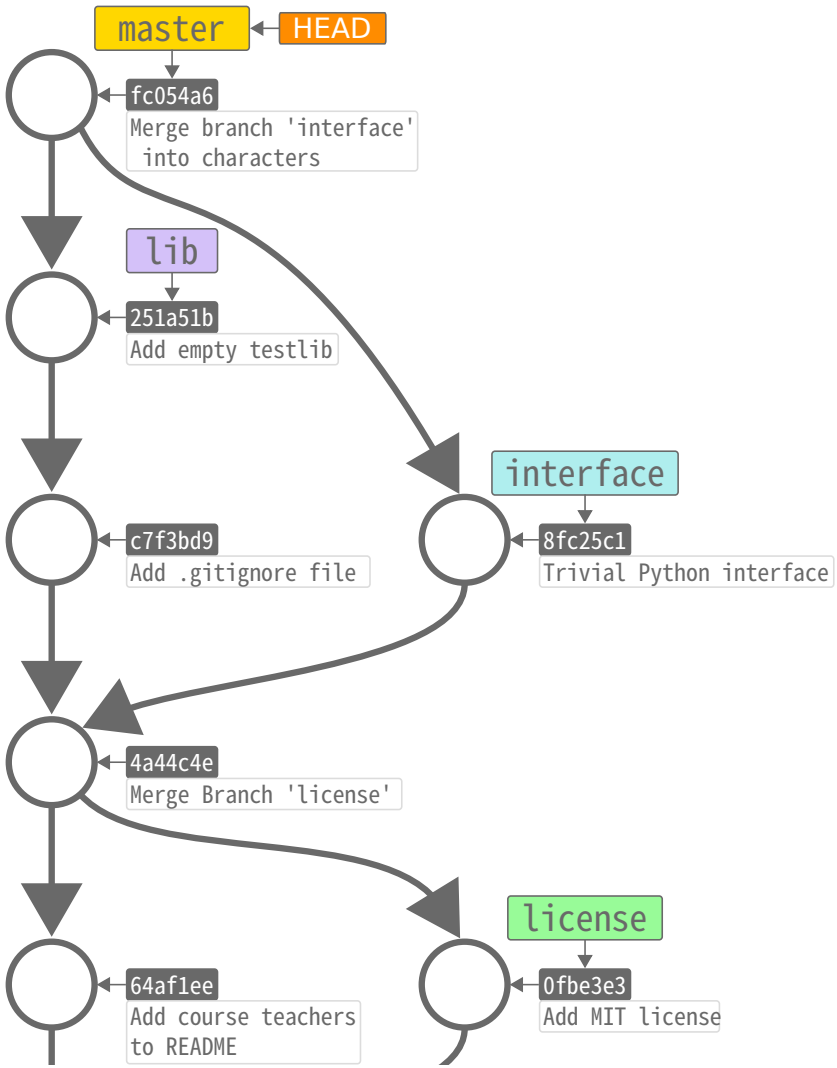
Manipulating the commit graph

Merged branch `interface` into `master`



Manipulating the commit graph

Merged branch `interface` into `master`



Note

Make sure that the working directory is clean before merging.

Otherwise:

```
$ git merge lib
error: Your local changes to the following files
       would be overwritten by merge:
        testlib.py
Please, commit your changes or stash them before
you can merge.
Aborting
```

Deleting branches

Old branches that have been merged can be deleted.

```
$ git branch -d interface
Deleted branch interface (was 8fc25c1).
```

```
$ git branch -d lib license
Deleted branch lib (was 251a51b).
Deleted branch license (was 0fbe3e3).
```

No history is lost, just labels removed.

Deleting branches

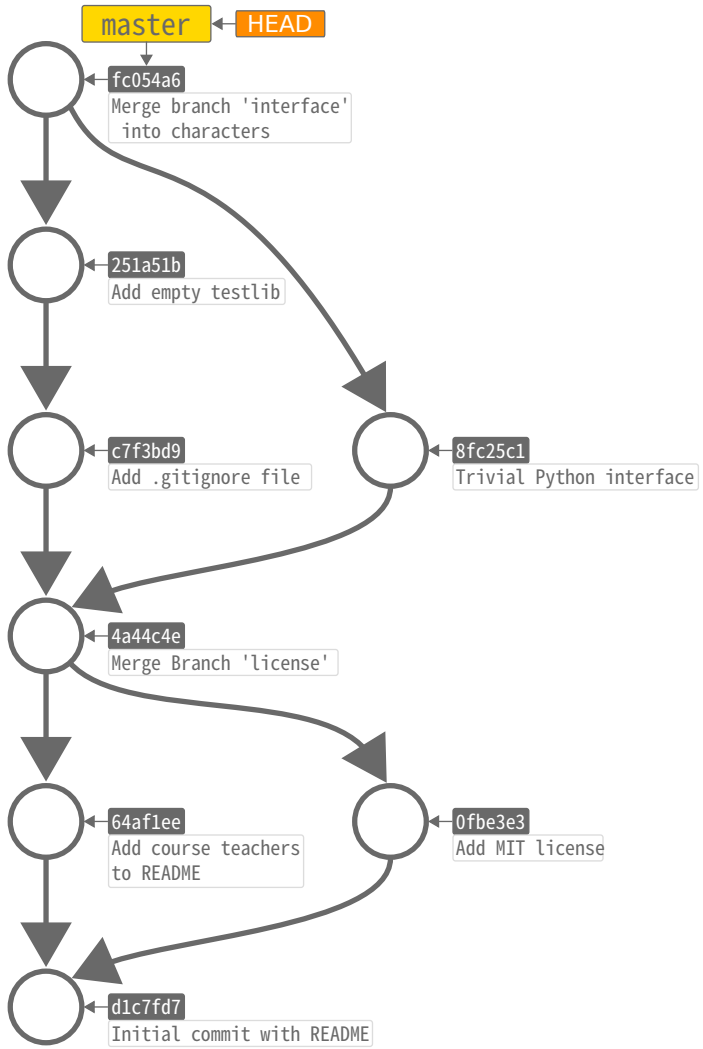


Table of contents

Merge conflicts

Git is pretty good at merging:

- The changes might have been in different files.
- Or in different parts of the same file.
- Git tries to figure out a sensible result.

Merge conflicts

Git is pretty good at merging:

- The changes might have been in different files.
- Or in different parts of the same file.
- Git tries to figure out a sensible result.

Sometimes this is not possible:

- The changes might be incompatible.
- When we try `git merge`, Git gives up.
- This situation is called a *merge conflict*.

Setting the stage (1/3)

```
$ git log --oneline --decorate --graph --all
* 5edaf08 (hue) State character ...
| * f1ef19c (HEAD, master) State character ...
|/
* 1f6d2ab Initial commit
```

We'd like to merge branch `hue` into `master`.

Setting the stage (2/3)

The last commit on `master`:

```
$ git show
f1ef19c State character preference
diff --git a/FACTS.md b/FACTS.md
index de15194..ef40359 100644
--- a/FACTS.md
+++ b/FACTS.md
@@ -1,2 +1,4 @@
 Facts about television series
 ==============================
+
+My favorite character is Eric Cartman.
```

Setting the stage (3/3)

The last commit on `hue`:

```
$ git show --oneline hue
5edaf08 State character preference
diff --git a/FACTS.md b/FACTS.md
index de15194..5e69508 100644
--- a/FACTS.md
+++ b/FACTS.md
@@ -1,2 +1,4 @@
 Facts about television series
 ==============================
+
+My favorite character is Milhouse.
```

Creating a merge conflict

```
$ git merge hue
Auto-merging FACTS.md
CONFLICT (content): Merge conflict in FACTS.md
Automatic merge failed; fix conflicts and then
commit the result.
```

Creating a merge conflict

```
$ git merge hue
Auto-merging FACTS.md
CONFLICT (content): Merge conflict in FACTS.md
Automatic merge failed; fix conflicts and then
commit the result.
$ git status
# On branch master
# Unmerged paths:
#   (use "git add/rm <file>.." as appropriate)
#    to mark resolution)
#                both modified:      FACTS.md
# no changes added to commit (use "git add" and/or
# "git commit -a")
```

Resolving a merge conflict (1/3)

```
$ cat FACTS.md
Facts about television series
==============================

<<<<<<< HEAD
My favorite character is Eric Cartman.
=======
My favorite character is Milhouse.
>>>>>>> hue
```

What we had is under HEAD, what hue had is above hue.

Resolving a merge conflict (2/3)

We resolve the conflict by hand.

```
$ nano FACTS.md
$ cat FACTS.md
Facts about television series
==============================

My favorite characters are Eric Cartman
and Milhouse.
```

Resolving a merge conflict (3/3)

And can now finish the merge commit.

```
$ git add FACTS.md
$ git commit
[master 1e496cb] Merge branch 'hue'
```

Aborting a merge

If you don't feel like resolving the merge conflict, you can go back with `git merge --abort`.

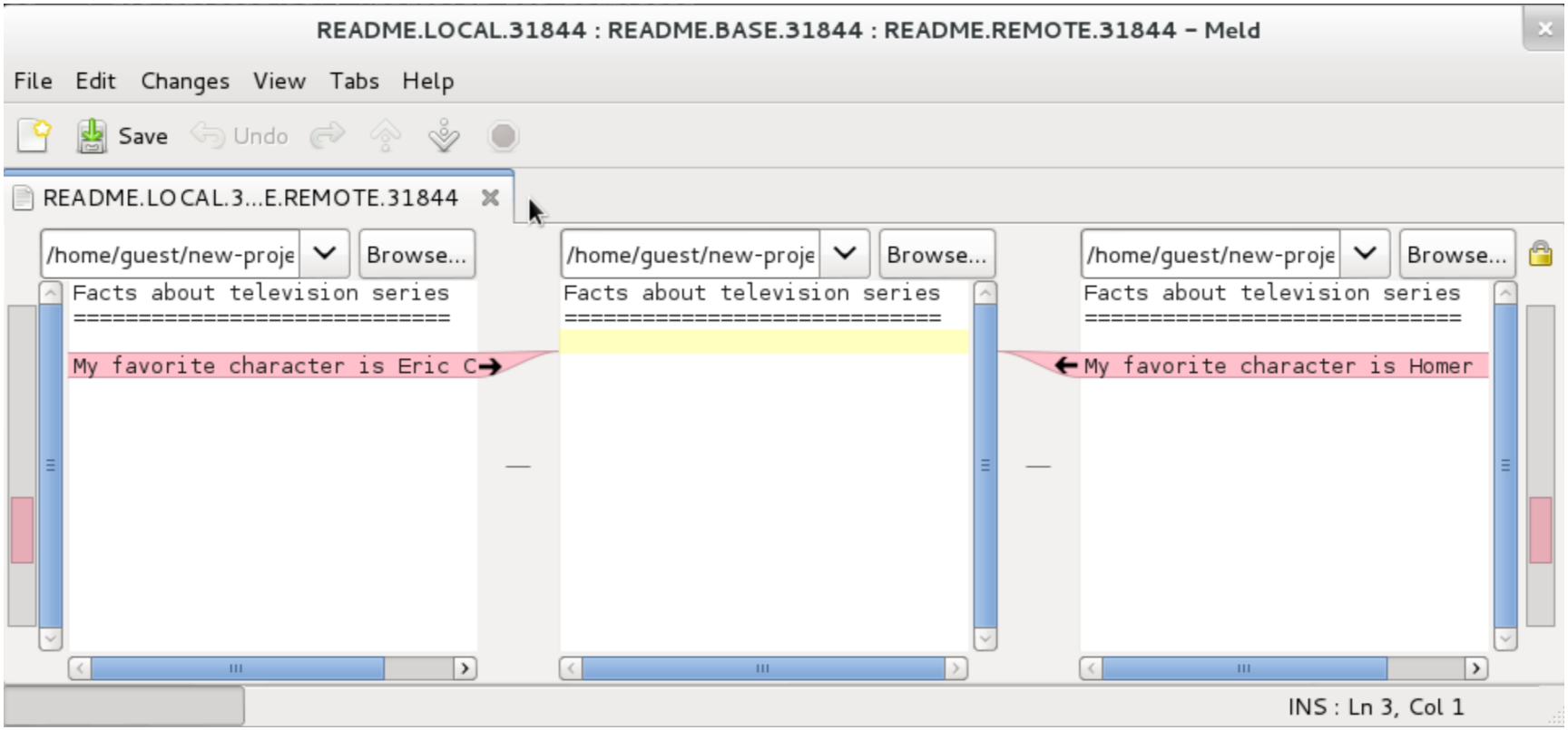
```
$ git merge hue
Auto-merging FACTS.md
CONFLICT (content): Merge conflict in FACTS.md
Automatic merge failed; fix conflicts and then
commit the result.
$ git merge --abort
$ git status
# On branch master
# nothing to commit (working directory clean)
```

Resolving conflicts with `git mergetool`

We can also use graphical merge tools such as *Meld*.

```
$ git mergetool
merge tool candidates: meld opendiff kdiff3 ...
Merging:
FACTS.md

Normal merge conflict for 'FACTS.md':
  local: modified file
  remote: modified file
Hit return to start merge resolution tool (meld):
```

Meld example



Tools like Meld provide an editable three-way diff.

Leiden University Medical Center

Acknowledgements

Martijn Vermaat
Wibowo Arindrarto
Szymon Kiełbasa
Jeroen Laros

`http://git-scm.com/book`
`https://www.atlassian.com/git`